การจัดเก็บข้อมูลของเอดิเตอร์

บุญเลิศ เอี่ยมทัศนา

ผมได้รับคำขอ ให้นำเรื่องราวของการใช้หน่วยความจำในฮีพที่จองไว้ด้วยคำสั่ง GetMem ในภาษาปาสคาลมากล่าวถึงบ้าง แต่ผมก็ลืมทุกครั้งไม่ว่าเมื่อเขียนหนังสือ "คู่มือเทอร์โบปาสคาล รุ่น 4.0-5.0" หรือเมื่อปรับปรุงหนังสือ "เรียนรู้ภาษาปาสคาล" ทั้งนี้เพราะไม่ค่อยได้นำหน่วยความจำนี้มาใช้

เมื่อคิดจะเขียนเรื่องนี้ก็มองหาโครงงานที่จะนำมาเป็นตัวอย่าง เห็นว่าหากนำเรื่องราวของโปรแกรมประเภทเอดิเตอร์ เช่นในไซด์คิดหรือในเทอร์โบปาสคาลมากล่าวถึงจะง่ายขึ้น เพราะเป็นที่รู้จักกันแพร่หลายทั่วไป และผมจะกล่าวถึงเฉพาะการจัดเก็บข้อมูลเท่านั้น คงไม่ต้องการคำอธิบายมากนัก

โปรแกรมเอดิเตอร์

หน้าที่อันสำคัญของโปรแกรมเอดิเตอร์เป็นไปดังที่ทราบกันอยู่คือ ให้ผู้ใช้สามารถป้อนข้อมูลในแบบ full screen edit ได้โดย ให้ผู้ใช้สามารถเลื่อนเคอร์เซอร์ไปได้ทั่วทั้งจอภาพ เพื่อการแทรก-ลบข้อมูลในแต่ละบรรทัด หรือแทรก-ลบ-ย้ายข้อมูลทั้งบรรทัด หรือหลาย ๆ บรรทัด ซึ่งการดำเนินการทั้งหมดนี้มีพื้นฐานอยู่ที่การจัดเก็บข้อมูลกับ การแทรก-ลบข้อมูลทั้งบรรทัด

ดังนั้น ในบทความของผมนี้จะกล่าวถึงเฉพาะ 2 เรื่องนี้เท่านั้น โดยจะกล่าวรวม ๆ กันไปคือ

ในตัวอย่างทุกตัวอย่างจะเริ่มโดยให้ผู้ใช้ป้อนข้อมูล 4 บรรทัด เก็บไว้ในหน่วยความจำ เสร็จแล้วให้นำออกแสดงซ้ำ แล้วให้ผู้ใช้เลือกแทรกข้อมูล ณ บรรทัดใดก็ได้ นำออกแสดงซ้ำอีก รายการสุดท้ายคือให้ผู้ใช้เลือกที่จะลบ คือยกเลิกข้อมูลที่บรรทัดใดก็ได้ แล้วนำผลออกแสดง และจะขอพูดเฉพาะภาษาอังกฤษเพราะถ้าเป็นภาษาไทยจะมีเรื่องอื่นอีกมาก

การจัดเก็บข้อมูล

ตามที่กล่าวแล้วว่า ในขั้นแรกผมตั้งใจว่าจะหาตัวอย่างเพื่อแสดงการใช้หน่วยความจำที่จองไว้ในฮีพ แต่เมื่อมาเลือกเอาการจัดเก็บข้อมูลของเอดิเตอร์เป็นตัวอย่าง ก็คงต้องพูดโดยถือการจัดเก็บข้อมูลเป็นหลักครับ

สำหรับการจัดเก็บข้อมูลของโปรแกรมเอดิเตอร์ที่ผมจะกล่าวถึงนี้ ผมไม่ได้นำมาจากที่ใด เพราะไม่ทราบว่าได้กล่าวไว้ ณ ที่ใดบ้าง (อาจมีแต่ผมไม่ทราบ) เพียงแต่คิดว่าน่าจะเป็นในรูปแบบใดบ้าง เท่าที่ผมคิดออกมีดังนี้ครับ

1. เก็บเป็นอะเรย์ (จำนวนบรรทัด) ของสตริง
2. เก็บเป็นอะเรย์ (จำนวนบรรทัด) ของพอยน์เตอร์ไปยังสตริงในฮีพ ซึ่งจะขอเรียกสั้น ๆ ว่าพอยน์เตอร์ไปยังสตริง
3. เก็บเป็นอะเรย์ (จำนวนบรรทัด) ของอินเด็กซ์ไปยังอะเรย์ของอักขระในฮีพ ซึ่งจะขอเรียกสั้น ๆ ว่าอินเด็กซ์ไปยังอะเรย์
4. เก็บเป็นดับเบิลลิงค์ลิสต์ (double link list)

โดยที่ผมจะขอนำมาแสดงเพียง 2 แบบเท่านั้นคือ พอยน์เตอร์ไปยังสตริง กับอินเด็กซ์ไปยังอะเรย์ ส่วนอีก 2 แบบจะขอนำไปกล่าวในตอนท้าย ว่าทำไมถึงไม่นำมาแสดง

พอยน์เตอร์ไปยังสตริง

การจัดเก็บข้อมูลในแบบนี้จะกำหนดเป็นอะเรย์ของพอยน์เตอร์ไปยังสตริง ขอให้ดูการกำหนดแบบข้อมูลและตัวแปรในรูป A1 จำนวนช่องของอะเรย์คือจำนวนบรรทัดตามแต่ผู้เขียนโปรแกรมจะกำหนด ได้กำหนดไว้เป็น 2000 ดังที่กำหนดให้แก่ค่าคงตัว MAXR

A1

```
PROGRAM  PointerToString(input,output);
USES     CRT;
CONST    MAXR = 2000;  MAXC = 80;  MAXCP = 82;
         CR = #13;  LF = #10;
TYPE     StrT  = String[MAXCP];
         PntrT = ^StrT;
         RowT  = ARRAY[1..MAXR] OF PntrT;
VAR      row    : RowT;
         r, rm : Word;
```

สำหรับความยาวของสตริง คือ จำนวนอักษรในแต่ละบรรทัด ขึ้นอยู่กับงาน สำหรับงานทางด้านเอกสาร (document) ส่วนมากจะเป็น 65 ถึง 80 อักษร จึงกำหนดไว้เป็น 80 อักษร ดังที่กำหนดค่าคงตัว MAXC

ได้กำหนดตัวแปร row เป็นอะเรย์ของพอยน์เตอร์ไปยังสตริงดังกล่าวมา เพื่อใช้ชี้ไปยังสตริงในฮีพอีกทอดหนึ่ง ในแบบ 1 พอยน์เตอร์ต่อ 1 สตริง ซึ่งในการเขียนโปรแกรมสามารถแยกออกได้เป็น 2 แนวความคิดคือ แบบแรกให้กำหนดหน่วยความจำในฮีพทั้ง 2000 สตริงในตอนต้น ซึ่งเหมาะสำหรับโปรแกรมที่เก็บข้อมูลแบบอื่นในฮีพด้วย

อีกแบบหนึ่งให้กำหนดสตริงในฮีพเมื่อต้องการใช้และยกเลิกเมื่อไม่ต้องการใช้เหมาะกับโปรแกรมที่ทำหน้าที่เอดิเตอร์แต่เพียงอย่างเดียว ซึ่งผมจะขอเลือกแบบหลังนี้เพราะเขียนโปรแกรมได้สั้นกว่า ง่ายกว่า สะดวกต่อการทำความเข้าใจ

A2

```
PROCEDURE  InsRow(n : Word);
var    i : Word;
BEGIN
  Inc(row);
  FOR  i := rm  DOWNTO n+1  DO  row[i] := row[i - 1];
  New(row[n]);
END;

PROCEDURE  DelRow(n : Word);
var    i : Word;
BEGIN
  Dispose(row[n]);  dec(rm);
  FOR  i := n TO rm  DO  row[i] := row[i + 1];
END;


PROCEDURE  DispData;
BEGIN
  WriteLn('===>');
  for  r := 1 to rm  do  WriteLn(row[r]^);
END;
```

A3

```
BEGIN
  ClrScr;  rm := 0;

  FOR  r := 1 TO 4  DO
    BEGIN  New(row[r]);   inc(rm);
           Write(r,' ');  ReadLn(row[r]^)  END;
  DispData;

  Write('Insert at line = ');  ReadLn(r);
  InsRow(r);
  Write('enter ');  ReadLn(row[r]^);
  DispData;

  Write('Delete line number = ');  ReadLn(r);
  DelRow(r);
  DispData;

writeln('end of program press  ENTER ');  Readln
END.
```

ตอนต้นของโปรแกรม หลักรูป A3 ได้ให้ผู้ใช้ป้อนข้อความ 4 บรรทัดเข้ามาเก็บไว้ จะเห็นได้ว่าได้กำหนดหน่วยความจำในฮีพครั้งละ 1 สตริง คือที่ใช้คำสั่ง New เสร็จแล้วได้ให้ผู้ใช้แทรกและถอนข้อมูลบรรทัดที่ต้องการ ในทุกขั้นตอนได้ให้แสดงข้อมูลที่เก็บไว้เพื่อแสดงว่าดำเนินการถูกต้องเพียงไร

ในโปรแกรมนี้ได้ใช้คำสั่ง ReadLn และ WriteLn กับสตริงในฮีพโดยตรง ซึ่งถ้าเป็นการดำเนินการจริงต้องเขียนโปรแกรมเพื่อดำเนินการในเรื่องนี้เพิ่มขึ้นมาอีก แต่ในบทความนี้ต้องการแสดงเพียงขั้นการถอนและแทรกข้อมูลทั้งบรรทัดว่าดำเนินการได้อย่างไรเท่านั้น กล่าวคือ

การแทรกบรรทัด

อยู่ในโพรซีเดอร์ InsRow รูป A2 จากการกำหนดแบบข้อมูล RowT ผมถือว่าช่องหนึ่ง ๆ คือบรรทัดหนึ่ง ๆ และนับว่าช่องที่ 1 คือ row[1] เป็นบรรทัดที่ 1 เมื่อผู้ใช้ต้องการแทรกบรรทัด จะหมายถึงเป็นการแทรกหน้าบรรทัดนั้น เช่น ต้องการแทรกบรรทัดที่ 3 จะเป็นการแทรกหน้าบรรทัดที่ 3 คือข้อความใหม่จะเป็นบรรทัดที่ 3 แทน ส่วนบรรทัดที่ 3 เดิมจะกลายเป็นบรรทัดที่ 4 ใหม่ และต่อ ๆ ไป

การแทรกจึงเป็นการเลื่อนข้อมูลในอะเรย์ลงล่าง 1 ช่องต่อ ๆ กันไป(เมื่อคิดว่าช่องที่ 1 อยู่บนสุดและช่องที่ MAXR อยู่ล่างสุด) แต่การเลื่อนจริงเลื่อนถึงช่องที่มีข้อมูลจริงที่อยู่ล่างสุด คือ rm เท่านั้น จะได้ว่าช่องที่ row[n] เป็นช่องว่างได้กำหนดสตริงในฮีพให้ด้วยคำสั่ง New เมื่อกลับไปโปรแกรมหลักได้ให้ผู้ใช้ป้อนข้อความมาเก็บในบรรทัดนี้

ดังผลการรันซึ่งรวมทั้งการลบข้อมูลทั้งบรรทัดด้วยในรูปที่ 1 ที่ให้

```
1 line one,the first line
2 line two
3 line three, the most favourable line.
4 line four, the 4th line
===>
line one,the first line
line two
line three, the most favourable line.
line four, the 4th line
Insert at line = 2
enter this is the insert line
===>
line one,the first line
this is the insert line
line two
line three, the most favourable line.
line four, the 4th line
Delete line number = 4
===>
line one,the first line
this is the insert line
line two
line four, the 4th line
end of program press  ENTER
```

รูปที่ 1 ผลการรันโปรแกรม PointerTo String

แสดงในแบบนี้ เพราะยังไม่ได้เขียนโปรแกรมดำเนินการกับจอภาพ

การลบบรรทัด

อยู่ในโพรซีเดอร์ DelRow รูป A2 ซึ่งจะลบคือ ยกเลิกบรรทัดที่ n เริ่มด้วยยกเลิกสตริงนี้ในฮีพและเลื่อนบรรทัดต่อไปมาแทนที่ ด้วยการเลื่อนข้อมูลในอะเรย์ช่องที่ n+1 มาแทนที่ n และต่อ ๆ ไป เช่นเดียวกับการแทรกบรรทัดแต่กลับทางกัน

โปรแกรมทั้งหมดที่เกี่ยวกับการจัดเก็บข้อมูลมีเพียงเท่านี้ จะเห็นว่าสามารถแทรกและลบบรรทัดได้โดยสะดวก ซึ่งผมจะขอยังไม่กล่าวถึงข้อดีข้อเสีย แต่มีเรื่องที่ขอกล่าวถึงก่อนคือ การกำหนดแบบข้อมูลเป็นอะเรย์ โดยให้แต่ละช่องแทน 1 บรรทัด เช่นที่กล่าวในเรื่องนี้และที่จะกล่าวในหัวข้อต่อไป เราจะต้องเสียอะเรย์ไป 1 ช่องต่อ 1 บรรทัด แม้แต่เป็นบรรทัดว่างคือเมื่อเว้นบรรทัด

มีเรื่องที่หาตัวอย่างได้ยากเห็นสมควรนำมากล่าวถึงคือ ถ้าเราต้องการดำเนินการกับสตริงในฐานะอะเรย์ เช่นต้องการอ้างถึงอักขระตัวที่ 8 ของบรรทัดที่ 3 สามารถดำเนินการได้ในฐานะอะเรย์ของอะเรย์ คือ

row [3]^[8]

อินเด็กซ์ไปยังอะเรย์ของอักขระ

การเก็บข้อมูลในแบบนี้ จะดำเนินการโดยกำหนดหน่วยความจำหน่วยหนึ่งเป็นผืนเดียวติดต่อกันในฮีพ เป็นอะเรย์ของอักขระ ขนาดของอะเรย์ตามแต่เราจะกำหนดแต่สำหรับเทอร์โบปาสคาลจะต้องไม่เกิน 65521 ไบต์ ดังการกำหนดแบบข้อมูล Database ในรูป B1

แต่เนื่องจากต้องการเก็บข้อมูลนี้ในฮีพจึงได้กำหนดแบบข้อมูล BaseT เป็นพอยน์เตอร์ นำไปกำหนดตัวแปร base ซึ่งในโปรแกรมหลักรูป B4 เมื่อใช้คำสั่ง GetMem (ต้นตอของเรื่องนี้) จะทำให้ base ชี้ไปที่อะเรย์ขนาด 65521 อักขระนี้ ซึ่งขอกำหนดเรียกเป็น Database ตามแบบข้อมูลของมัน

ในการแบ่งเป็นบรรทัด เนื่องจาก Database เป็นอะเรย์เราจึงสามารถอ้างถึงอักขระในแต่ละช่องได้ด้วยอินเด็กซ์ และเนื่องจากมีขนาดไม่ถึง 64 กิโลไบต์ จึงสามารถใช้ข้อมูลแบบ Word เป็นอินเด็กซ์ได้

ดังนั้น ถ้าผมกำหนดตัวแปรเป็นอะเรย์ของ Word โดยให้แต่ละช่องของอะเรย์ใหม่นี้เป็นอินเด็กซ์ไปยัง Database ผมก็สามารถแบ่ง Database ออกเป็นท่อน ๆ ได้ โดยที่ท่อนหนึ่ง ๆ หมายถึงบรรทัดหนึ่ง ๆ จะมีอักษรกี่ตัวก็ได้ (ตั้งแต่ 0 ถึง 65521 ตัว) ขึ้นอยู่กับการใช้งานของผู้ใช้ ดังที่กำหนดแบบข้อมูล RowT และตัวแปร row โดยให้มีบรรทัดได้ไม่เกิน 2000 บรรทัด (เฉลี่ยบรรทัดละ 32 อักษร)

การทำงานของโปรแกรมหลักคงเป็นไปตามที่กล่าวมา แต่ไม่สามารถใช้คำสั่ง ReadLn และ WriteLn ได้ จึงได้สร้างโพรซีเดอร์ ReadRow และ WriteRow รูป B2 ขึ้นมาดำเนินการแทน กล่าวคือ

B1

```
PROGRAM  IndexToArray(input,output);
uses  crt;
CONST  MAXHEEP = 65521;  MAXR = 2000;
       CR = #13;  LF = #10;
TYPE   Database = ARRAY[1..MAXHEEP] OF Char;
       BaseT = ^Database;
       RowT  = ARRAY[1..MAXR] OF Word;
VAR    base : baseT;  last : Word;
       row  : RowT;   r,rm   : Word;  ch : Char;
```

ReadRow

จะอ่านอักขระตรงจากคีย์บอร์ดบัฟเฟอร์เข้ามาครั้งละ 1 ตัว นำไปเก็บใน Database เริ่มจากช่องที่โปรแกรมหลักส่งมาให้ไปจนกว่า ผู้ใช้กดคีย์ Enter ซึ่งจะได้รหัส 13 คือ CR (carriage return) มา นำไปเก็บและนำออกจอภาพ ทำให้เคอร์เซอร์เลื่อนไปอยู่ที่ต้นบรรทัด จึงให้ปิดท้ายอะเรย์ท่อนนี้ด้วยรหัส 10 คือ LF (line feed) เมื่อนำออกจอภาพจะทำให้เคอร์เซอร์ขึ้นบรรทัดใหม่ (โปรแกรมนี้ยังไม่สมบูรณ์เพราะยังไม่สามารถให้แทรกหรือลบข้อมูลในแต่ละบรรทัดได้)

หากดูในโปรแกรมหลักรูป B4 จะเห็นว่า เริ่มด้วยให้ค่า 1 แก่ตัวแปร last ซึ่งผมได้ตั้งใจไว้ว่าจะใช้ตัวแปรนี้ชี้ที่ท้ายข้อมูลใน Database (ตรงช่องที่จะเก็บอักขระตัวต่อไป) และให้ rm (คือย่อมาจาก row max) มีค่าเป็น 0 โดยตั้งใจไว้ว่าจะให้ rm ชี้ช่องสุดท้ายในอะเรย์ row ที่มีข้อมูล เทียบได้กับบรรทัดที่ เช่นเดียวกับที่กล่าวมาในหัวข้อที่แล้ว

แต่การดำเนินการต่างกันเช่น ในขั้นตอนที่ให้ผู้ใช้ป้อนข้อมูล จะเท่ากับเริ่มโดยให้ row[1] ชี้ไปที่ต้น Database เมื่อ ReadRow อ่านข้อมูลเข้ามาเก็บใน Database จะทำให้ค่าของ last เพิ่มขึ้นเท่ากับจำนวนอักษรที่ผู้ใช้ป้อนบวกด้วย CR และ LF ซึ่งขอกำหนดเรียกเป็นสตริง (ไม่ใช่สตริงในภาษาปาสคาล) เมื่อเริ่มวงรอบที่ 2 ในวงรอบ FOR จะเท่ากับให้ row[2] ชี้ที่ท้ายสตริงแรกและเป็นต้นสตริงที่ 2

ดังนั้นเมื่อจบการทำงานในวงรอบ FOR จะได้ว่า row [1] ถึง row [4] ชี้ไปที่ต้นสตริงที่ 1 ถึงที่ 4 หรือคือบรรทัดที่ 1 ถึงที่ 4 นั่นเอง โดยที่แต่ละสตริงจะปิดท้ายด้วย CR และ LF และแต่ละสตริงอาจมีความยาวไม่เท่ากันตามแต่ผู้ใช้จะป้อน ทำให้ประหยัดหน่วยความจำได้มากกว่าที่จะเก็บในแบบอะเรย์ของสตริงแต่ขึ้นอยู่กับลักษณะของงานด้วย

WriteRow

หลังจากที่ให้ผู้ใช้ป้อนข้อความ 4 บรรทัดแล้ว ได้ให้นำออกแสดงด้วย DispData จะเห็นว่าใน DispData (รูป

B2

```
PROCEDURE  ReadRow(var id : Word);
BEGIN
  REPEAT
    ch := ReadKey;  Write(ch);
    base^[id] := ch;  Inc(id);
  UNTIL  ch = CR;
  Write(LF);  base^[id] := LF;  Inc(id);
END;

PROCEDURE  WriteRow(id : Word);
BEGIN
  REPEAT
    Write(base^[id]);  Inc(id);
  UNTIL  (base^[id - 1] = LF);
END;

PROCEDURE  DispData;
BEGIN
  WriteLn('===>');
  for  r := 1 to rm  do  WriteRow(row[r])
END;
```

B4

```
BEGIN   { MAIN }
  GetMem(base,MAXHEEP);  base^[1] := CR;  base^[2] := LF;
  ClrScr;  last := 1;  rm := 0;

  WriteLn('enter 4 string');
  FOR  r := 1 TO 4  DO
    BEGIN    row[r] := last;  ReadRow(last);  inc(rm)  END;
  DispData;

  Write('Insert line number? ');  ReadLn(r);  inc(rm);
  InsRow(r);  row[r] := last;
  Write('enter text:  ');  ReadRow(last);  DispData;

  Write('Delete line number? ');  ReadLn(r);  dec(rm);
  DelRow(r);  DispData;

write('end of program press  ENTER ');  ReadLn;
END.
```

B2) ได้ส่งอินเด็กซ์ที่ชี้ไปยังแต่ละต้นสตริง ซึ่งเก็บไว้ในอะเรย์ rows ไปให้ WriteRow

ใน WriteRow จึงเพียงแต่นำค่าใน Database เริ่มจากช่องนี้ออกจอภาพจนสุดสตริง คือเมื่อตรวจพบอักขระ LF

InsRow

การดำเนินการต่อไปในโปรแกรมหลัก คือให้ผู้ใช้เลือกแทรกข้อมูลหน้าบรรทัดใดก็ได้ โดยอ่านมาเก็บไว้ในตัวแปร r และเพิ่มจำนวนบรรทัดคือ rm ขึ้นหนึ่ง แล้วเรียกใช้ InsRow เพื่อการแทรกบรรทัด

การดำเนินการของ InsRow รูป B3 ดำเนินการเช่นเดียวกับที่กล่าวในหัวข้อที่แล้วคือ เลื่อนข้อมูลในอะเรย์ลงล่าง 1 บรรทัด (ขอให้สังเกตการใช้ตัวแปร n เทียบกับการใช้ตัวแปร i ในโปรแกรม PointerToSring)

ส่วนข้อมูลที่ผู้ใช้ป้อนใหม่ให้นำไปต่อท้ายข้อมูลเดิมใน Database โดยให้อินเด็กซ์ในบรรทัดที่แทรกใหม่นี้ชี้ไปยังต้นสตริงนี้ (การดำเนินการอยู่ในโปรแกรมหลัก)

DelRow

สำหรับการลบบรรทัดของโปรแกรมนี้คือดำเนินการในโพรซีเดอร์ DelRow (รูป B3) แตกต่างจากในโปรแกรม PointerToString เพราะต้องเลื่อนบรรทัดใน row ด้วย เลื่อนข้อมูลใน Database เพื่อลบข้อมูลในบรรทัดนี้ด้วย และต้องปรับค่าอินเด็กซ์ที่ชี้ไปยังแต่ละสตริงด้วย

จึงดำเนินการโดยเลื่อนข้อมูลใน Database ไปทางต้นอะเรย์ก่อน คือโปรแกรมท่อนแรก ส่วนโปรแกรมท่อนหลังเป็นการยกเลิกบรรทัดในอะเรย์ row ด้วยการเลื่อนบรรทัดขึ้นบน 1 บรรทัด เช่นเดียวกับที่กล่าวมา เสร็จแล้วให้ปรับค่าอินเด็กซ์ใหม่ เท่ากับระยะทางที่ข้อมูลของตนถูกเลื่อนไป ซึ่งต้องดำเนินการตั้งแต่ row[1] เพราะผู้ใช้อาจจะแทรกบรรทัดที่ 1 ไว้ก่อนหน้านี้ ซึ่งข้อมูลจะอยู่ท้าย Database ทำให้ถูกเลื่อนไปด้วย

นำมาพิจารณา

จากแบบข้อมูล 2 แบบที่กล่าวมานี้ และผมขอกล่าวรวมทั้ง 4 แบบตามที่กล่าวในตอนต้นด้วย สามารถนำมาพิจารณาได้ ดังนี้

พอยน์เตอร์ไปยังสตริง

การเก็บข้อมูลในแบบนี้เหมาะกับงานทางด้านเอกสาร เช่นบทความ เพราะจะมีบรรทัดว่างหรือบรรทัดที่มีอักษรไม่เต็มบรรทัดอยู่น้อยมาก จึงไม่มีปัญหาเรื่องการประหยัดหน่วยความจำ คือไม่เป็นข้อเสีย

แต่ข้อดีที่เหนือกว่าแบบอินเด็กซ์ไปยังอะเรย์คือ เขียนโปรแกรมได้ง่ายกว่ากันมาก และไม่ต้องเสียเวลาในการเคลื่อนย้ายข้อมูลเมื่อลบบรรทัด นอกจากนี้ยังใช้หน่วยความจำในฮีพ คือที่เหลือจากโปรแกรมได้ทั้งหมด เช่น อาจจะเหลือถึง 300 กิโลไบต์

อินเด็กซ์ไปยังอะเรย์

การเก็บข้อมูลในแบบนี้เหมาะกับงานที่ผู้ใช้มักไม่ป้อนข้อมูลจนเต็มบรร-

B3

```
PROCEDURE  InsRow(n : Word);
BEGIN
  IF  rm <= MAXR  THEN
    for  n := rm downto n+1  do  row[n] := row[n-1];
END;


FUNCTION  LengthRow(id : Word) : Word;
VAR   i : Byte;
BEGIN
  i := 0;
  while  base^[id] <> LF  do  begin  inc(i);  inc(id)  end;
  LengthRow := i + 1;
END;

PROCEDURE  DelRow(n : Word);
VAR   len : Byte;  i,stt : Word;
BEGIN
  stt := row[n];  len := LengthRow(stt);  last := last - len;
  for  i := stt to last  do  base^[i] := base^[i + len];
  for  i := n to rm  do  row[i] := row[i + 1];
  for  i := 1 to rm  do
    if  row[i] >= stt  then  row[i] := row[i] - len;
END;
```

ทัด เช่นซอร์สโปรแกรมของภาษาต่าง ๆ จะประหยัดหน่วยความจำได้มาก

ส่วนข้อเสียคือต้องเขียนโปรแกรมยาวกว่า ต้องเคลื่อนย้ายข้อมูลทุกครั้งที่ลบบรรทัด และสำหรับเทอร์โบปาสคาลบนเครื่อง IBM-PC คือจะเก็บข้อมูลได้ไม่เกิน 65521 อักขระ

อะเรย์ของสตริง

การเก็บข้อมูลในแบบนี้ ก็คือเก็บในแบบอะเรย์สองมิตินั่นเอง แต่เป็นการนำเอาข้อเสียของทั้งสองแบบมารวมกัน กล่าวคือ เหมาะสำหรับข้อมูลแบบเอกสารเท่านั้น ต้องย้ายข้อมูลทุกครั้งที่ลบบรรทัดและจำนวนข้อมูลมากกว่า (หากผู้ใช้ป้อนข้อมูลไม่เต็มบรรทัดเป็นจำนวนมาก) และสำหรับเทอร์โบปาสคาลบนเครื่อง IBM-PC จะเก็บข้อมูลได้ไม่ถึง 64 กิโลไบต์เช่นกัน เป็นแบบที่ไม่เหมาะอย่างยิ่งที่จะนำมาใช้จึงไม่ได้นำมากล่าว

ดับเบิลลิงค์ลิสต์

การเก็บข้อมูลในแบบนี้ตัวข้อมูลอาจเก็บอยู่ในรูปของสตริง มีพอยน์เตอร์ชี้ไปยังข้อมูลตัวก่อน คือบรรทัดก่อน 1 ตัว และข้อมูลตัวต่อไป คือบรรทัดต่อไปอีก 1 ตัว ดังรูปที่ 2 จึงคล้ายแบบพอยน์เตอร์ไปยังสตริงมาก ข้อดีและข้อเสียจึงขอเทียบกับแบบพอยน์เตอร์ไปยังสตริง กล่าวคือ

ทางด้านข้อดี ไม่ต้องกำหนดจำนวนบรรทัดไว้ตายตัว (ดังเช่นที่กำหนดให้แก่ row) จำนวนบรรทัดจะเพิ่มขึ้นหรือลดลง ตามแต่ผู้ใช้จะป้อนแทรกหรือลบบรรทัด

ส่วนข้อเสียคือ ไม่สามารถอ้างข้อมูลในแต่ละบรรทัดได้โดยตรงในแบบอะเรย์ ต้องดำเนินการในแบบลิงค์ลิสต์ คือต้องใช้พอยน์เตอร์อีก 1 ตัว ไล่ไปมาในแบบซีเควนเชียล ทำให้ต้องเขียนโปรแกรมยาวกว่า ทำงานช้ากว่า เมื่อลบล้างกันแล้วเห็นว่ามีข้อเสียมากกว่า จึงไม่ได้นำมากล่าวถึง

ดังนั้น จึงเห็นว่ามีแบบข้อมูลให้เลือกเพียง 2 แบบ การที่จะเลือกแบบใดขึ้นอยู่กับงานคือ ถ้าเป็นเอดิเตอร์อิสระ เช่น WordStar ก็น่าจะเลือกใช้แบบพอยน์เตอร์ไปยังสตริง แต่ถ้าถูกจำกัดด้วยเนื้อที่ของหน่วยความจำ เช่น เอดิเตอร์ของเทอร์โบปาสคาล ก็ควรจะใช้แบบอินเด็กซ์ไปยังอะเรย์ (แต่ไม่ได้หมายความว่าโปรแกรมทั้งสองนี้จะเก็บข้อมูลตามแบบที่ผมกล่าวมาหรือไม่)

เกี่ยวกับภาษาไทย

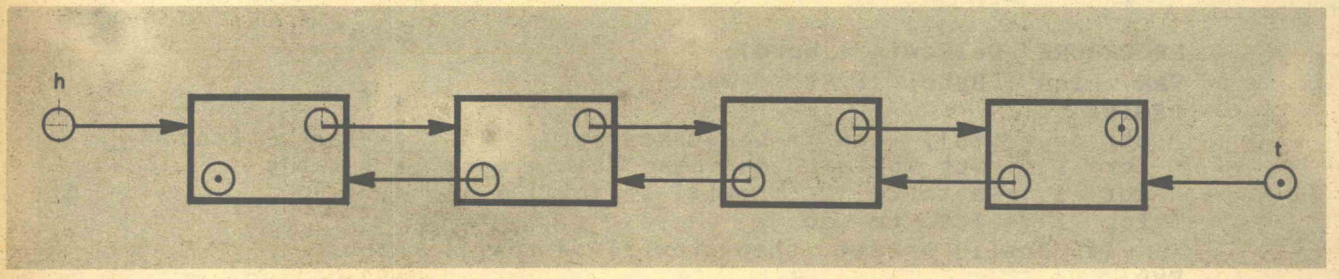
แต่สำหรับภาษาไทยแล้วผมเห็นว่าการเก็บข้อมูลในแบบอินเด็กซ์ไปยังอะเรย์จะประหยัดหน่วยความจำได้มากที่สุด ไม่ว่าจะใช้ในงานใด ทั้งนี้เพราะว่าในภาษาอังกฤษ ถ้ากำหนดไว้บรรทัดละ 80 อักขระ เมื่อมีข้อมูลเต็มบรรทัดก็จะเป็น 80 อักษรพอดี แต่ในภาษาไทยไม่ใช่ กล่าวคือ

อาจมีเพียง 80 อักษร (เช่นพิมพ์อักษร ก ทั้งบรรทัด ซึ่งเป็นไปไม่ได้) หรืออาจมีถึง 240 อักษร (เช่น พิมพ์คำว่า กู๋ ทั้งบรรทัด ซึ่งเป็นไปไม่ได้อีกเช่นกัน) มีผู้ทำสถิติไว้ว่าไม่เกิน 140 อักษร แต่ใครจะกล้ารับประกัน หากเชื่อตามนี้ก็ต้องเชื่อต่อไปด้วยว่าจะมีข้อมูลเฉลี่ยบรรทัดละประมาณ 100 อักษร หากกำหนดไว้บรรทัดละ 140 อักษร ก็จะเสียหน่วยความจำไปเปล่า ๆ ถึงบรรทัดละประมาณ 40 ไบต์ ซึ่งถ้าเก็บในแบบอินเด็กซ์ไปยังอะเรย์จะแก้ปัญหานี้ได้ทั้งหมด

ข้อเสียของการเก็บข้อมูลแบบอินเด็กซ์ไปยังอะเรย์ที่สำคัญคือ ในเรื่องการเคลื่อนย้ายข้อมูลขอให้ป้อนโปรแกรมในรูปที่ 3 แทนโปรแกรมในรูป B4 คือป้อนไว้หน้าโปรแกรมหลักในรูป B4 แล้วทดลองรัน

โปรแกรมนี้จะเคลื่อนย้ายข้อมูลจำนวน 65520 ไบต์ ซึ่งเป็นจำนวนมากที่สุดที่จะเคลื่อนย้ายไปทางต้น Database หลังจากกดคีย์ Enter แล้ว และจะแสดงอักขระ * เมื่อเลื่อนเสร็จ จะเห็นว่านานพอสมควร (สำหรับเครื่อง 4.77 MHz) จึงอยู่ว่าจะยอมรับหรือไม่ยอมรับ และในการดำเนินการจริงไม่ค่อยมีโอกาสที่จะเคลื่อนย้ายข้อมูลมากมายขนาดนี้ กับมีเรื่องที่จะปรับปรุงได้ดังจะกล่าวต่อไป

อนึ่ง ระยะทางที่ย้ายคือจะย้ายข้อมูลกี่อักขระ เช่นตามตัวอย่างที่ยกมานี้เป็นการย้ายข้าม 1 อักขระ หากระยะทางมากกว่านี้ เช่นเมื่อลบบรรทัดที่ผู้ใช้ป้อนข้อมูลไว้ 50 อักขระ ก็จะเป็นการย้ายข้าม 50 อักขระ ไม่ทำให้



รูปที่ 2 ข้อมูลแบบดับเบิ้ลลิงค์ลิสต์

```
var  i : Word;
begin
  GetMem(base,MAXHEEP);  base^[1] := CR;  base^[2] := LF;
  ClrScr;  last := 1;  rm := 0;

  Write('press ENTER: ');  ch := Readkey;
  for  i := 1 to MAXHEEP - 1  do base^[i] := base^[i + 1];
  WriteLn('*');
end.
```

รูปที่ 3 โปรแกรมหลักที่ใช้ทดลองจับเวลาในการเคลื่อนย้ายข้อมูล 65520 ไบต์

```
VAR     base,base1,base2 : baseT;


   GetMem(base1,MAXHEEP);
   GetMem(base2,MAXHEEP);


   if  (rm > MAXR) or (last > MAXHEEP)  then  base := base2
   else  base := base1;
```

รูปที่ 4 ส่วนของโปรแกรมที่แสดงการจองและการใช้หน่วยความจำในฮีพหลายหน่วย

เวลาในการเคลื่อนย้ายข้อมูลเพิ่มขึ้น มีแต่ลดลงเพราะจำนวนข้อมูลที่ย้ายลดลง

ส่วนเรื่องข้อจำกัดที่สามารถเก็บข้อมูลได้ไม่ถึง 64 กิโลไบต์นั้น สามารถแก้ได้โดยจองหน่วยความจำแบบ base ไว้หลาย ๆ หน่วย ดังแสดงในรูปที่ 4 เป็นส่วนของโปรแกรมที่เพียงแสดงให้เห็นว่าจะดำเนินการได้อย่างไรคือ โปรแกรมส่วนอื่น ๆ คงติดต่อกับ Database ผ่าน base โดยมีโปรแกรมส่วนหนึ่งคอยเปลี่ยนให้ base แทน base1 หรือ base2

ในทำนองเดียวกันนี้ หากเห็นว่าการเคลื่อนย้ายข้อมูลดำเนินการได้ช้า ก็อาจแบ่งดาต้าเบสออกเป็นเบสย่อยหลาย ๆ เบส เบสละประมาณ 1 หมื่นไบต์ ผู้ใช้จะไม่สังเกตเห็นช่วงเวลาที่ใช้ในการเคลื่อนย้ายข้อมูล

ปรับปรุงแบบพอยน์เตอร์ไปยังสตริง

หากคุณไม่ยอมรับแบบการจัดเก็บข้อมูลในแบบอินเด็กซ์ไปยังอะเรย์ ก็มีวิธีที่จะปรับปรุงแบบพอยน์เตอร์ไปยังสตริงคือ ตัดความยาวของสตริงที่คิดว่าไม่ควรเกิน 140 อักษร และที่คิดว่าเฉลี่ยบรรทัดละประมาณ 100 อักษร เป็น 2 ท่อน ท่อนละ 55 ตัวอักษร (เผื่อขาดไว้บ้าง) ดังการกำหนดแบบข้อมูลและตัวแปรในรูปที่ 5

```
CONST  MAXC = 55;  MAXR = 2000;
TYPE   StrT  = String[MAXC];
       PntrT = ^OneSect;
       OneSect = RECORD
          data : StrT;
          next : PntrT;
                 END;
       RowT = ARRAY[1..MAXR] OF PntrT;
VAR    row : RowT;
```

รูปที่ 5 แสดงการกำหนดแบบข้อมูลและตัวแปรในการเก็บข้อมูลในแบบพอยน์เตอร์ไปยังสตริงที่ต่อกันแบบลิงค์ลิสต์

การใช้งานจะเริ่มโดยใช้หน่วยความจำ 1 ท่อน (One Sect) ก่อนด้วย New เมื่อผู้ใช้ป้อนข้อมูลเกิน 55 อักษร ก็จะใช้ท่อนที่ 2 และให้ next ของท่อนแรกชี้ไปยังท่อนที่สองนี้ ดังนั้นในบรรทัดปกติจะใช้หน่วยความจำเพียง 2 ท่อน หากเป็นบรรทัดที่มีสระบน-ล่าง และวรรณยุกต์มากกว่าปกติ ก็จะใช้ท่อนที่ 3 รวมเป็น 3 ท่อน หากผิดปกติก็จะใช้ท่อนต่อ ๆ ไป

แบบผสม

หากคุณยังไม่พอใจอีกก็มีวิธีการที่จะปรับปรุงแบบการเก็บข้อมูล เป็นแบบผสมระหว่างพอยน์เตอร์ไปยังสตริงกับอินเด็กซ์ไปยังอะเรย์ ซึ่งจะขอเรียกว่าแบบพอยน์เตอร์ไปยังอะเรย์ กล่าวคือ

ในแบบอินเด็กซ์ไปยังอะเรย์ ตรงที่ผมใช้คำสั่ง GetMem นั้น ความจริงตรงนั้นใช้คำสั่งว่า New (base) ได้เช่นกัน ไม่ได้เป็นการใช้ประโยชน์ของคำสั่ง GetMem แต่อย่างใด ซึ่งความจริงแล้วคำสั่ง GetMem เปิดโอกาสให้เราจองเนื้อที่ในหน่วยความจำตามแบบข้อมูลที่ต้องการ แต่ไม่ได้ขึ้นอยู่กับขนาดของแบบข้อมูลนั้น หากขึ้นอยู่กับที่เราต้องการ เช่น

หากเรากำหนดแบบข้อมูลอะเรย์ของอักขระขนาด 100 ไบต์ เราอาจจะจองหน่วยความจำเพียง 10 ไบต์ หรือ 100 หรือ 10000 ไบต์ ได้ทั้งนั้น

ดังนั้นหากเราอ่านข้อมูลเข้ามาเก็บในบัฟเฟอร์ ซึ่งอาจเป็นอะเรย์หรือสตริงก่อน ตรวจว่ายาวเท่าไรแล้วจึงจองหน่วยความจำในฮีพตามนี้ จากนี้จึงถ่ายทอดข้อมูลจากบัฟเฟอร์ไปเก็บในฮีพ เมื่อผู้ใช้ต้องการยกเลิกข้อมูลก็เพียงแต่ใช้คำสั่ง FreeMem โดยไม่ต้องเคลื่อนย้ายข้อมูล เป็นการรวมข้อดีของทั้งสองแบบไว้ด้วยกัน

ขอให้ดูการกำหนดแบบข้อมูลและตัวแปรในรูป C1 จะเห็นว่าได้กำหนดขนาดของอะเรย์คือ StrT ไว้เพียง 10 ไบต์ เพื่อแสดงให้เห็นว่าทำได้ สำหรับตัวเลขที่เหมาะสมน่าจะเป็น 240 ไบต์ เพื่อบอกแก่ผู้อ่านโปรแกรมว่าบรรทัดหนึ่ง ๆ จะไม่เกินนี้ และได้กำหนดตัวแปร buf ให้เป็นสตริงขนาด 255 อักขระ ตัวเลขนี้ก็เพียงแต่กำหนดให้มากไว้เท่านั้น

ตัวแปรอีกตัวหนึ่งที่กำหนดขึ้นใหม่คือ len เพื่อเก็บความยาวของอะเรย์ในแต่ละบรรทัด เพราะเราต้องใช้ค่าความยาวนี้ในที่หลายแห่ง

ในด้านการอ่านข้อมูลเข้ามาเก็บใช้โพรซีเดอร์ ReadRow รูป C2 ซึ่งคงใช้คำสั่ง ReadLn (ในโปรแกรมจริงต้องเขียนใหม่) อ่านข้อมูลที่ผู้ใช้ป้อน

C1

```
PROGRAM  PointerToArray(input,output);
USES     CRT;
CONST    MAXR = 2000;  MAXC = 10;
         CR = #13;  LF = #10;
TYPE     StrT  = ARRAY[1..MAXC] OF Char;
         PntrT = ^StrT;
         RowT  = ARRAY[1..MAXR] OF PntrT;
VAR      row   : RowT;  buf : String;
         r, rm : Word;
         len : ARRAY[1..MAXR] OF Word;
```

C2

```
PROCEDURE  ReadRow(n : Word);
VAR  i : Word;
BEGIN
  ReadLn(buf); len[n] := Length(buf) + 2;
  GetMem(row[n],len[n]);
  for  i := 1 to len[n] - 2  do  row[n]^[i] := buf[i];
  inc(i);  row[n]^[i] := CR;  inc(i);  row[n]^[i] := LF;
END;


PROCEDURE  WriteRow(n : Word);
var  i : Byte;
BEGIN
  i := 1;
  REPEAT
    Write(row[n]^[i]);  inc(i);
  UNTIL  row[n]^[i - 1] = LF;
END;
```

C3

```
PROCEDURE  InsRow(n : Word);
var    i : Word;
BEGIN
  Inc(rm);
  FOR  i := rm  DOWNTO n+1  DO  row[i] := row[i - 1];
END;

PROCEDURE  DelRow(n : Word);
var    i : Word;
BEGIN
  FreeMem(row[n],len[n]);  dec(rm);
  FOR  i := n TO rm  DO  row[i] := row[i + 1];
END;

PROCEDURE  DispData;
BEGIN
  WriteLn('===>');
  for  r := 1 to rm  do
    WriteRow(r);
END;
```

C4

```
BEGIN
  ClrScr;  rm := 0;

  FOR  r := 1 TO 4  DO
    BEGIN  Write(r,' ');  ReadRow(r);  inc(rm)  END;
  DispData;

  Write('Insert at line = ');  ReadLn(r);
  InsRow(r);
  Write('enter ');  ReadRow(r);
  DispData;

  Write('Delete line number = ');  ReadLn(r);
  DelRow(r);
  DispData;

writeln('end of program press  ENTER ');  Readln
END.
```

มาเก็บให้บัฟเฟอร์ buf หาความยาวเก็บไว้ในอะเรย์ len แล้วจึงจองหน่วยความจำ ลอกข้อมูลจากบัฟเฟอร์ไปไว้ในฮีฟดังที่กล่าวมา แล้วปิดท้ายด้วย CR กับ LF สำหรับโปรแกรมท่อนอื่นตามรูป C3 และ C4 คงเป็นไปตามที่กล่าวมาต่างกัน ในเรื่องของการดำเนินการเท่านั้น

สำหรับข้อเสียของการเก็บข้อมูลแบบนี้คือ ที่ต้องอ่านข้อมูลเข้ามาก่อนแล้วย้ายไปเก็บในฮีพอีกครั้งหนึ่งนั้น ผู้ใช้จะไม่รู้สึกเลยคงเป็นข้อเสียเฉพาะต้องเขียนโปรแกรมยาวขึ้นเท่านั้น แต่มีอีกเรื่องหนึ่งคือหากผู้ใช้ยกเลิกบรรทัดเล็กบรรทัดน้อยมาก ๆ อาจทำให้ฮีพเต็มได้

ตามที่กล่าวมานี้เป็นเพียงแค่เสนอการจัดเก็บข้อมูลในแบบต่าง ๆ เท่านั้น โปรแกรมที่กล่าวมาก็เป็นเพียงโปรแกรมเพื่อการทดลอง เมื่อนำไปใช้จริงต้องปรับปรุงและต่อเติมอีกมาก ส่วนการที่จะเลือกการจัดเก็บข้อมูลแบบใดก็สุดแต่จะพิจารณาครับ